Finding a Bounded-Degree Expander Inside a Dense One

Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, Luca Trevisan

Finding a Bounded-Degree Expander Inside a Dense One

Luca Becchetti*
Sapienza Università di Roma
Rome, Italy
becchetti@dis.uniroma1.it

Andrea Clementi
Università di Roma Tor Vergata
Rome, Italy
clementi@mat.uniroma2.it

Emanuele Natale
Université Côte d'Azur, CNRS, INRIA
Sophia Antipolis, France
natale@unice.fr

Francesco Pasquale†
Università di Roma Tor Vergata
Rome, Italy
pasquale@mat.uniroma2.it

Luca Trevisan‡
Bocconi University
Milan, Italy
l.trevisan@unibocconi.it

Abstract

It follows from the Marcus-Spielman-Srivastava proof of the Kadison-Singer conjecture that if $G = (V, E)$ is a Δ-regular dense expander then there is an edge-induced subgraph $H = (V, E_H)$ of G of constant maximum degree which is also an expander. As with other consequences of the MSS theorem, it is not clear how one would explicitly construct such a subgraph.

We show that such a subgraph (although with quantitatively weaker expansion and near-regularity properties than those predicted by MSS) can be constructed with high probability in linear time, via a simple algorithm. Our algorithm allows a distributed implementation that runs in $\mathcal{O}(\log n)$ rounds and does $\mathcal{O}(n)$ total work with high probability.

The analysis of the algorithm is complicated by the complex dependencies that arise between edges and between choices made in different

*Partially supported by ERC Advanced Grant 788893 AMDROMA "Algorithmic and Mechanism Design Research in Online Markets" and MIUR PRIN project ALGADIMAR "Algorithms, Games, and Digital Markets".

†Partially supported by the University of "Tor Vergata" under research programme "Mission: Sustainability" project ISIDE (grant no. E81I18000110005).

‡LT was supported by the NSF under grant CCF 1815434 and his work on this project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 834861).

rounds. We sidestep these difficulties by following the combinatorial approach of counting the number of possible random choices of the algorithm which lead to failure. We do so by a compression argument showing that such random choices can be encoded with a non-trivial compression.

Our algorithm bears some similarity to the way agents construct a communication graph in a peer-to-peer network, and, in the bipartite case, to the way agents select servers in blockchain protocols.

1 Introduction

The proof of the Kadison-Singer conjecture by Marcus, Spielman and Srivastava [16] (henceforth, the *MSS Theorem*) has several important graph theoretic corollaries. In particular, if $G = (V, E)$ is an undirected graph with n nodes in which every edge has effective resistance $\mathcal{O}(n/|E|)$, then there is an edge-induced subgraph $H = (V, E_H)$ of G that has $\mathcal{O}(n/\varepsilon^2)$ edges and that is an unweighted ε-spectral-sparsifier[1] of G.

Interesting examples of graphs to which this statement applies are edge-transitive graphs, such as the hypercube, and regular expanders of constant normalized edge expansion. As with other consequences of the MSS Theorem, and other non-constructive results proved with similar techniques, it is not known how to construct such subgraphs in polynomial (or even subexponential) time.

In the case of regular expanders, the result, qualitatively, states that if $G = (V, E)$ is a Δ-regular graph of constant normalized edge expansion, there exists an edge-induced subgraph H of G that has constant maximum degree and constant normalized edge expansion.

In this work, we show how to constructively find such an H, assuming that $\Delta = \Omega(n)$ and that the second eigenvalue of the adjacency matrix of G (which measures the spectral expansion of the graph) is at most a sufficiently small constant times the degree Δ. The randomized algorithm we propose receives as input a Δ-regular graph G and two integer parameters d and c.

If we only assume $\Delta = \Theta(n)$, $c > 2n/|E|$ and d is a sufficiently large absolute constant then, with high probability, the algorithm completes in $\mathcal{O}(n)$ steps and returns a subgraph H of G, in which each node has degree between d and $(c+1)\cdot d$ (see Theorem 4).

If we further assume that the second eigenvalue of the adjacency matrix of G is at most $\gamma\Delta$, with γ a sufficiently small constant, we can prove that, with high probability, H has conductance $\Omega(1)$ (see Theorem 5).

Our algorithm is extremely simple and naturally lends itself to a distributed implementation, in a model in which the underlying communication network is G itself, with its nodes as computing elements. In this model, the nodes of G

[1] A weighted graph $H = (V, E_H)$ is an ε-spectral-sparsifier [2] of a graph $G = (V, E_G)$ if, for every vector $\mathbf{x} \in \mathbb{R}^V$, we have $(1-\varepsilon)\sum_{(u,v)\in E_G}(x_u - x_v)^2 \leqslant \sum_{(u,v)\in E_H} w_H(u,v)\cdot(x_u - x_v)^2 \leqslant (1+\varepsilon)\sum_{(u,v)\in E_G}(x_u - x_v)^2$ where $w_H(u,v)$ is the weight of the edge (u,v) in H. We say that H is *unweighted* if the weights of all the edges of H are all equal to the same scaling factor $|E_G|/|E_H|$.

can collectively identify a subgraph H with the properties mentioned above in $\mathcal{O}(\log n)$ rounds and with $\mathcal{O}(n)$ total work and communication cost, in the sense that at the end of the protocol, each node knows its neighbors in H.

The distributed version of our algorithm, that we call RAES (for *Request a link, then Accept if Enough Space*), works in rounds, each consisting of two phases. Initially, each node has 0 outgoing links and 0 incoming links. In the first phase of each round, each node v selects enough random neighbors (according to the topology of G) so that linking to all of them would secure v a total of d outgoing links. It then submits a request to each selected neighbor to establish a link. In the second phase of the round, each node accepts all requests received in the first phase of the current round, unless doing so would cause it to exceed the limit of cd incoming links; if this is the case, the node rejects all requests it received in the first phase of the current round. The algorithm completes when each node has established exactly d outgoing links, so that no further requests are submitted. A formal description of the algorithm is given in Section 2.

To show that our algorithm completes in $\mathcal{O}(\log n)$ rounds with high probability when G is Δ-regular and $c > 2n/\Delta$, we show that, for any request submitted by some node v in any round t, regardless of the remaining randomness of the algorithm, the request is accepted with probability at least $1/2$. This happens since, in each round, the number of nodes that reject any request is at most $n/2$. This is enough to show that convergence takes $\mathcal{O}(\log n)$ rounds with high probability and total work $\mathcal{O}(dn)$ on average. To prove that the total work is $\mathcal{O}(dn)$ with high probability we show that, in each round t, if $\mathrm{d}_v^{\mathrm{out}}$ denotes the current number of v's outgoing links, $d \cdot n - \mathbf{E}\left[\sum_v \mathrm{d}_v^{\mathrm{out}}\right]$, i.e., the expected number of "missing links", shrinks, on average, by a constant factor. Moreover, the amount by which the above quantity changes at each step is a Lipschitz function of independent random variables, which means that we can argue with high probability about the amount by which this quantity decreases.

The main result of this work is the proof that, if G is a sufficiently good expander, then the graph produced by the algorithm has constant expansion. In the spirit of how one analyzes the expansion of random regular graphs, we would like to argue that, for every set $S \subseteq V$ of $s \leqslant n/2$ vertices, there is at least a probability, say, $1 - n^{-2} \cdot \binom{n}{s}^{-1}$, that, of the ds outgoing links from the vertices of S, at least $\Omega(ds)$ are links from S to $V - S$. Then we could use a union bound over all possible sets S to say that with probability at least $1 - 1/n$ every set S has at least $\Omega(ds)$ links crossing the cut and going into $V - S$. The probability distribution of the links created by the algorithm, and the ways in which they are correlated, are however very difficult to analyze.

Our approach is to use a *compression argument*: we show that the random choices of the algorithm that lead to a non-expanding graph can be non-trivially compressed, and hence have low probability. The approach of proving that an event is unlikely by showing that the random choices leading to it are compressible is often a convenient way to analyze the outcome of an algorithm. Such arguments are sometimes expressed in the language of *Kolmogorov complexity* [15] and they are often used in cryptography to analyze the security

of protocols that involve a random oracle, following [8]. In [19], the authors review various probabilistic analyses that can be performed using compression argument (which they call *encoding* arguments).

Our argument is roughly as follows: suppose that, in the graph constructed by the algorithm, S is a non-expanding set of vertices. If G is a sufficiently good Δ-regular expander, then, from the expander mixing lemma, we get that the typical vertex of S has only about $\Delta \cdot |S|/n$ neighbors in S, but, if S is non-expanding in H, then the typical node in S has, say, at least $.9 \cdot d$ of its d outgoing links in S. This means that, for the typical node in S, we can represent $.9d$ of its d outgoing links using $\log \frac{\Delta \cdot |S|}{n}$ bits instead of $\log \Delta$, with a saving of order of $d|S| \log \frac{n}{|S|}$ bits. For sufficiently large constant d, this is more than the $\log \binom{n}{|S|}$ bits that it takes to represent the set S. Unfortunately, things are not so easy because we need the representations of choices made by the nodes in the algorithm to be prefix-free, in order for their concatenation to be decodable. Therefore, we have to spend some additional bits in the representation of various terms, in particular for the choices that lead to links from S to $V - S$ (which are not so many since S is a non-expanding set) and for requests that are rejected. To complete the argument, we have to argue that the overall number of requests from nodes in S that are rejected cannot be too large, for we would otherwise have a non-trivial way of compressing their description. This is true because, as argued above, each request has a small probability of being rejected, so that realizations of the random algorithm that lead to many rejected requests are unlikely, hence compressible (for further details see Section 4.1).

Algorithm RAES is inspired by the way nodes create bounded-degree overlay networks in real-life distributed systems, such as peer-to-peer protocols [9, 20] like BitTorrent, or in distributed ledger protocols such as Bitcoin [21]. In this protocol for example, each node in a communication network is aware of the existence of a certain subset of the other nodes (in our algorithm, for the generic node v this subset corresponds to the set of v's neighbors in G). Each node tries to establish a minimum number of connections to other nodes (or to special "server" nodes[2]) and does so by selecting them at random from its list of known nodes (or known servers). Nodes also have a maximum number of connections they are going to accept, rejecting further connections once this limit is reached.

On the other hand, our algorithm does not capture important traits of peer-to-peer and blockchain models, such as the fact that nodes can join or leave the network, and that nodes can exchange their lists of known nodes, so that the graph "G" in fact is dynamic. We believe, however, that our analysis addresses important aspects, such as the complicated dependencies that arise between different links in the virtual network, and the *expansion* properties of the resulting virtual network. Expansion in particular is closely related to resilience to nodes leaving the network, a very important property in practice.

[2]In this setting, we notice that if G is bipartite then H is bipartite as well.

1.1 Related work

Distributed constructions of expanders.

Our main result is an efficient, distributed algorithm to construct a bounded-degree expander. This question has been addressed for a number of models and initial conditions. In [12], Law and Siu provide a distributed protocol running on the LOCAL asynchronous model that form expander graphs of arbitrary fixed degree d. Their goal is to maintain the expansion property under insertions, starting from a constant-size graphs, and they show how to do so in constant time and constant message complexity per node insertion. See also [9] and [22] for such sequential constructions of expanders.

In [1], Allen-Zhu et al. show a simple and local protocol that, starting from any connected d-regular connected graph with $d = \Omega(\log n)$, returns a d-regular expander. At every every round, an edge e is selected u.i.r. together with one length-3 path including e and, then, a suitable flipping of the edges of this path is performed (so, the obtained graph is not guaranteed to be a subgraph of the original graph). Their spectral analysis of the evolving graph shows that, after $\mathcal{O}(n^2 d^2 \text{polylog}\,(n)))$ rounds, the obtained random graphs is an expander, with high probability. Their algorithm models the way in which nodes exchange neighborhood information in real-life protocols, and it works starting from much more limited information than ours (their initial information is an arbitrary graph of logarithmic degree, while we start from a graph of linear degree which is already an expander), and the price they pay is a polynomial, rather than logarithmic, convergence time.

Sparsification.

We motivated our main result as a constructive proof of a special case of the sparsification results implied by the MSS theorem, for which no constructive proofs are known. Here it matters that we are interested in sparsifying a regular graph by using an unweighted subgraph of bounded maximum degree. If we allowed weighted graphs, and we were only concerned about the average degree of the sparsifier, then an explicit construction of constant average-degree sparsifiers for all graphs is given by the BSS sparsifiers of [2]. A parallel construction of the BSS sparsifier, however, is not known. Parallel construction of (weighted, unbounded max degree) sparsifiers have been studied [11], but such constructions involve graphs of logarithmic average degree, a setting in which our problem is trivial: given a Δ-regular graph $G = (V, E)$, if we choose each edge independently with probability order of $(\log n)/\Delta$, we get a graph that with high probability has maximum degree $\mathcal{O}(\log n)$ and, using matrix Chernoff bound, we can show that it is a spectral sparsifier of G if G is such that every edge has effective resistance $\mathcal{O}(n/|E|)$, including the case of expanders and of edge-transitive graphs.

Finally, in [7], Frieze and Molloy consider the task of partitioning expander graphs. In more detail, they provide a partitioning algorithm that, given as input a Δ-regular graph with edge-expansion Φ and a parameter k, returns a

partition $(E_1, \ldots, E_k)$ of the edges, such that each induced graph $G_i = (V, E_i)$ is almost-regular with node degree $\Theta(\frac{\Delta}{k})$ and it has edge expansion $\Omega(\Phi/k)$. Their algorithm runs in $\mathcal{O}(n^{\log \Delta})$ time and the required assumptions on Δ, k and ϕ do not allow to produce *constant*-degree subgraphs (their construction in fact requires $k = \mathcal{O}(\Delta/\log \Delta)$).

Parallel Balls-Into-Bins Processes.

If the underlying network is the complete graph K_n, then RAES can be seen as a *parallel balls-into-bins* algorithm [18, 13] with $m = dn$ balls, each one representing an outgoing-link request which must be assigned to one of n bins, corresponding to the nodes of the network. In this perspective, our algorithm assigns each ball to one bin, so that the maximum load of the bins is at most cd, for some constant c and the algorithm terminates in $\mathcal{O}(\log n)$ rounds with high probability. Several algorithms have been introduced for this problem and the best algorithms achieve constant maximum load within a constant number of rounds by using $k > 1$ random choices at every round for each ball [13]. The RAES strategy adopted by our algorithm is similar to the one used in the basic version of Algorithm PARALLELTHRESHOLD analysed in [3] by Berenbrink et Al, which is in turn a parallelized version of the scheduling strategy studied in [4]. They show that the convergence time is $\mathcal{O}(n \log m)$ when $cd = d + 1$, while our analysis implies that it is $\mathcal{O}(\log m)$ when c is an absolute constant larger than 1. The maximum number of balls accepted by each bin, called the *threshold*, is fixed to $\lceil m/n \rceil + 1$. They show this basic version, achieving an almost tight maximum load, converges within $\mathcal{O}(n \log m)$ rounds, w.h.p. They also conjecture a tight lower bound on the convergence time.

2 Preliminaries and main result

For an undirected graph $G = (V, E)$ of n nodes, we denote its adjacency matrix as A and the *volume* of a subset of nodes $U \subseteq V$ as $vol(U) = \sum_{u \in U} d_u$. Notice that when G is Δ-regular, we have $vol(U) = \Delta|U|$. Consider two (not necessarily disjoint) subsets $U, W \subseteq V$, we define $e(U, W)$ as the number of edges in G with one endpoint in U and the other in W.

Definition 1. *A graph $G = (V, E)$ is an ε-expander if, for every subset $U \subset V$ with $|U| \leqslant n/2$, the number $e(U, V - U)$ of edges in the cut $(U, V - U)$ is at least $\varepsilon \cdot vol(U)$.*

The expansion properties we derive for the subgraph returned by Algorithm RAES turn out to depend on the spectral gap of the input graph. In particular, our analysis uses the following "one-sided" version of the *Expander Mixing Lemma* [14], which establishes a connection between the second largest eigenvalue of the adjacency matrix A of G and its expansion properties and also holds for bipartite graphs.

Lemma 2. *Assume $G = (V, E)$ is a Δ-regular graph and let λ be the second largest eigenvalue of*[3] *A. Let S be any subset of nodes. Then, the number $e(S, S)$ of edges of G with both endpoints in S is at most*

$$\frac{1}{2}\left(\frac{\Delta|S|^2}{n} + \lambda|S|\right).$$

Proof. If 1_S is the indicator vector of S, then, it holds that

$$1_S^\intercal A 1_S = \sum_{v \in V} |N_v(S)| = 2 \cdot e(S, S),$$

where $N_v(S)$ is the set of v's neighbors in S and $e(S, S)$ is the number of edges with both end-points in S. Observe that the matrix $A - \Delta J/n$ (where J is the matrix having all entries set to 1) has largest eigenvalue λ, so we get

$$1_S^\intercal (A - \Delta J/n) 1_S \leqslant \lambda \|1_S\|^2 = \lambda \cdot |S|.$$

We also notice that $1_S(\Delta J/n)1_S^\intercal = \Delta|S|^2/n$. It thus follows that

$$e(S, S) \leqslant \frac{1}{2}\left(\frac{\Delta|S|^2}{n} + \lambda|S|\right).$$

□

In the next sections, we analyze the behaviour of Algorithm RAES on dense, regular expanders. The algorithm was informally described in the introduction, a more formal description is given below.

[3] Since G is Δ-regular, the bounds on λ we derive immediately translate into bounds on the second largest eigenvalue of G's normalized matrix.

Algorithm 1 RAES(G, d, c)

H := empty directed graph over the node set V
while H has nodes of outdegree $< d$ **do**
  PHASE 1: ▷ $\mathrm{d}_v^{\mathrm{out}}$: current outdegree of v in H
  for each node $v \in V$ **do**
    $v \in V$ picks $d - \mathrm{d}_v^{\mathrm{out}}$ neighbors in G uniformly at random
    v submits a connection request to each of them
  end for
  PHASE 2: ▷ $\mathrm{d}_v^{\mathrm{in}}$: current indegree of v in H
  for each node $v \in V$ **do**
    if v received $\leqslant cd - \mathrm{d}_v^{\mathrm{in}}$ connection requests in the previous phase **then**
      v accepts all of them and the corresponding directed links are added to H
    else
      v rejects all connection requests received in Phase 1
    end if
  end for
end while
Replace each directed link by an undirected one
return H

We next define the class of almost-regular graphs RAES stabilizes on w.h.p.

Definition 3. *A graph $G = (V, E)$ is a (d, cd)-almost regular graph if the degree d_v of any node $v \in V$ is such that $d_v \in \{d, \ldots, (c+1)d\}$.*

Our main results can be formally stated as follows.

Theorem 4. *For every $d \geqslant 1$, every $0 < \alpha \leqslant 1$, every $c \geqslant 2/\alpha$, and for every Δ-regular graph $G = (V, E)$ with $\Delta = \alpha n$, the time complexity of* RAES(G, d, c) *is $\mathcal{O}(n)$ w.h.p. Moreover, the algorithm can be implemented in the uniform* GOSSIP *distributed model*[4] *so that its parallel completion time is $\mathcal{O}(\log n)$ and its overall message complexity is $\mathcal{O}(n)$, w.h.p.*

Theorem 5. *A sufficiently small constant $\varepsilon > 0$ exists such that, for any constants $d \geqslant 44$ and $0 < \alpha \leqslant 1$, for any sufficiently large c*[5]*, and every Δ-regular graph $G = (V, E)$ with $\Delta = \alpha n$ and second largest eigenvalue of the adjacency matrix*[6] *$\lambda \leqslant \varepsilon\alpha^2\Delta$,* RAES$(G, d, c)$ *returns a (d, cd)-almost regular ε-expander $H = (V, A)$, w.h.p.*

[4] In this very-restrictive communication model [5, 10], at every synchronous step, each node can (only) contact a *constant* number of its neighbors, chosen u.i.r., and exchange two messages (one for each direction) with each of them.

[5] We didn't try to optimize the constants in our analysis, which shows that $c \geqslant \max\{(\frac{2}{\alpha})^2, 10e^{10d}\}$ suffices.

[6] I.e., G is a sufficiently good expander. Also note that, equivalently, we are imposing that the second largest eigenvalue of G's normalized adjacency matrix be at most $\varepsilon\alpha^2$.

The proof of Theorem 4 is given in Section 3, while the proof of Theorem 5, which is our main technical contribution, is described in Section 4.

3 Proof of Theorem 4

Throughout this section, we consider a Δ-regular graph $G = (V, E)$ with $\Delta = \alpha n$ for some arbitrary constant $0 < \alpha < 1$. We analyze the execution of Algorithm RAES on input G for any constants $d \geqslant 1$ and $c > 1/\alpha$.

Recall that, according to the process defined by RAES, each node v asks for d *link requests* to its neighbors and has cd *slots* to accomodate link requests from its neighbors.

We first provide a simple proof that RAES on input $G = (V, E)$ terminates within a logarithmic number of rounds[7], w.h.p.

Lemma 6. *For every $d \geqslant 1$, every $c > 1/\alpha$, and every $\beta > 1$, RAES(G, d, c) completes the task within $\beta \log(n)/\log(\alpha c)$ rounds, with probability at least $1 - d/n^{\beta-1}$.*

Proof. Let us fix an arbitrary ordering of the nd required links and, for $i = 1, \ldots, nd$, let $X_i^{(t)}$ be the binary random variable taking value 1 if link i is settled at the end of round t and 0 otherwise. First note that, since a link is settled at some round t if it was already settled at previous round $t-1$, it holds that

$$\mathbf{P}\left(X_i^{(t)} = 0\right) = \mathbf{P}\left(X_i^{(t)} = 0 | X_i^{(t-1)} = 0\right) \mathbf{P}\left(X_i^{(t-1)} = 0\right). \tag{1}$$

Let us name $\mathbf{Y}_{-i}^{(t)} = \left(Y_1^{(t)}, \ldots, Y_{i-1}^{(t)}, Y_{i+1}^{(t)}, \ldots, Y_{nd}^{(t)}\right)$ the random vector where, for each $j \neq i$, random variable $Y_j^{(t)}$ indicates the destination node of link j at round t. Observe that for every vector $\mathbf{y}_{-i} = (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_{nd}) \in V^{nd-1}$ it holds that

$$\mathbf{P}\left(X_i^{(t)} = 0 \mid X_i^{(t-1)} = 0,\, \mathbf{Y}_{-i}^{(t)} = \mathbf{y}_{-i}\right) \leqslant \frac{1}{\alpha c}. \tag{2}$$

Indeed, given any $\mathbf{y}_{-i} \in V^{nd-1}$, there are always at most $nd/(cd) = n/c$ nodes with cd or more incoming link requests. Hence, among the αn neighbors of the node asking link i, at least $(\alpha - 1/c)n$ have less than cd incoming requests. Hence, the probability that link i settles is at least $1 - 1/(\alpha c)$. Since (2) holds for any choice of $\mathbf{y}_{-i} \in V^{nd-1}$, we get that $\mathbf{P}\left(X_i^{(t)} = 0 \mid X_i^{(t-1)} = 0\right) \leqslant 1/(\alpha c)$ and thus from (1) we have that $\mathbf{P}\left(X_i^{(t)} = 0\right) \leqslant 1/(\alpha c)^t$. The thesis then follows from a union bound over all the nd links and from the fact that $t \geqslant \beta \log(n)/\log(\alpha c)$. □

[7] Notice that the meaning of *round* here is exactly that defined in the pseudocode of RAES.

Remark. The first proof we gave for the above lemma was based on a simple compression argument [19]. We describe it in Appendix B since it can be used by the reader as a "warm-up" for the more difficult analysis given in Section 4.

The time complexity of Algorithm RAES is asymptotically bounded by the total number of link requests produced by its execution on graph $G = (V, E)$. Lemma 6 easily implies that this number is $\mathcal{O}(dn \log n)$, w.h.p. In the next lemma we prove a tight $\mathcal{O}(nd)$ bound.

Lemma 7. *For every constants $d \geqslant 1$ and $c > 2/\alpha$, the total number of link requests made by* RAES(G, d, c) *(and thus the time complexity) is $\Theta(n)$, w.h.p.*

Proof. Let us fix an arbitrary ordering of the nd required links and, for $i = 1, \ldots, nd$, let $Z_i^{(t)}$ be the binary random variable taking value 1 if link i is not yet settled at the beginning of round t and 0 otherwise. The random variable indicating the total number of link requests produced by the algorithm can thus be written as

$$Z = \sum_{t=0}^{\infty} \sum_{i=1}^{nd} Z_i^{(t)} .$$

Proceeding as in the proof of Lemma 6, it is easy to see that for every $t \in \mathbb{N}$ it holds that

$$\mathbf{E}\left[\sum_{i=1}^{nd} Z_i^{(t)}\right] \leqslant \frac{nd}{(\alpha c)^t} .$$

Hence, the total expected number of link requests is $\mathbf{E}[Z] \leqslant \frac{\alpha c}{\alpha c - 1}\, nd$.

In order to prove that $Z = \mathcal{O}(nd)$ w.h.p., we first show that whenever the number of unsettled links is above $nd/\log n$, it decreses by a constant factor, w.h.p. Formally, for any $k \geqslant nd/\log n$, we derive the following inequality

$$\mathbf{P}\left(\sum_{i=1}^{nd} Z_i^{(t)} > \frac{k}{\alpha c/2} \,\middle|\, \sum_{i=1}^{nd} Z_i^{(t-1)} = k\right) \leqslant e^{-\frac{k}{2\alpha^2 c^4 d^2}} . \tag{3}$$

Notice that random variables $Z_i^{(t)}$ conditional on the graph formed by the links settled at the end of round $t-1$ are not independent, so we cannot use a standard Chernoff bound. However, we can use the *method of bounded differences* [6, Corollary 5.2] (see Theorem 16 in Appendix A), since the sum of the $Z_i^{(t)}$ conditional on the graph of the $nd - k$ settled links at the end of the previous round can be written as a $2cd$-Lipschitz function of the independent k random variables indicating the link requests at round t.

In more details, we name $u^{(t-1)}$ the set of k unsettled links at the end of round $t-1$ and consider random variables $\{Y_i\}_{i \in u^{(t-1)}}$, each of them returning the node-destination index that the non-assigned link request i tries to connect to. Observe that Y_i's are mutually independent and, moreover, the sum in (3) can be written as a deterministic function of them: $\sum_{i=1}^{nd} Z_i^{(t)} = f(Y_{i_1}, \ldots, Y_{i_k})$.

Moreover, this function is $2cd$-Lipschitz w.r.t. its arguments: If we change one of the arguments Y_i, we are moving a request i from a node v_1 to a node v_2. The largest impact this can have on $\sum_{i=1}^{nd} Z_i^{(t)}$ is that the response for each of all the link requests sent to v_2 changes. However, if this number was already larger than cd, then the moving of link request i would not have any impact. This means that, in the worst-case, at most cd link requests trying to connect to v_2 switch from assigned to non-assigned. At the same time, a symmetric argument holds for the link requests trying to connect to v_1. In formulas, for all vectors of nodes $\left(v_{i_1}, \ldots, v_{i_j}, \ldots, v_{i_k}\right)$ and $\left(v_{i_1}, \ldots, v^\star_{i_j}, \ldots, v_{i_k}\right)$ differing only on a single entry i_j, it holds that

$$\left| f\left(v_{i_1}, \ldots, v_{i_j}, \ldots, v_{i_k}\right) - f\left(v_{i_1}, \ldots, v^\star_{i_j}, \ldots, v_{i_k}\right) \right| \leqslant 2cd\,.$$

Therefore, by applying Corollary 5.2 in [6] (see also Theorem 16 in Appendix A), with $\mu \leqslant M = k/(\alpha c)$ and $\beta_j = 2cd$ for all $j = 1, \ldots, k$ we get (3).

From (3) and the chain rule, it follows that, for $T = \mathcal{O}\left(\frac{\log\log n}{\log(\alpha c/2)}\right)$ rounds, the number of unassigned link requests decreases by a factor $\alpha c/2 > 1$ at each round, w.h.p., until it becomes smaller than $nd/\log n$. These rounds thus account for $nd\sum_{t=0}^{T}\left(\frac{2}{\alpha c}\right)^t = \mathcal{O}(nd)$ connection requests, w.h.p. Then, from Lemma 6 it follows that the remaining $\frac{nd}{\log n}$ link requests are assigned within $\mathcal{O}(\log n)$ rounds, w.h.p., thus accounting for at most further $\mathcal{O}(nd)$ additional link requests, w.h.p. □

Distributed implementation.

As one can easily verify from its pseudocode, Algorithm RAES is designed to work over any synchronous parallel distributed model where the nodes of the input graph $G = (V, E)$ are the local computing units which can communicate via the bidirectional links defined by the set of edges E. We remark that, at every round, each node contacts (i.e. sends link requests to) only a constant number of its neighbors. It thus follows that RAES induces a decentralized protocol that can be implemented on the communication-constrained *uniform* GOSSIP model [5, 10]. Notice that the protocol does not require any global labeling of the nodes, rather, it requires that each node knows some local labeling of its bi-directional ports.

In this setting, Lemma 6 easily implies that every node completes all of its tasks within $\Theta(\log n)$ rounds, w.h.p.

As for communication complexity, we observe that all the point-to-point communications made by the protocol can be encoded with 1-bit messages (accept/reject the link request). Moreover, Lemma 7 implies that the overall number of links requests (and thus of exchanged messages) is w.h.p. $\Theta(dn)$, which is clearly a tight bound for this task.

Finally, we notice that if nodes know an upper bound n' on n, since G is regular, then they can locally derive a sufficiently good lower bound of α, i.e., $\alpha' = \alpha/\mathrm{poly}(n)$. Then, by Lemma 6, after round $T = 2\log(n')/\log(\alpha' c)$, every node can decide to stop any action (so it terminates) and it will be aware that the protocol has completed the global task, w.h.p.

4 Proof of Theorem 5

In the previous subsection, we showed that, after $T = \mathcal{O}(\log n)$ rounds, Algorithm RAES stabilizes to a subgraph $H = (V, E_H)$ of the input graph $G = (V, E)$ that turns out to be a (d, cd)-almost regular graph. In this Section, we provide the proof of Theorem 5: we indeed show that if G is an expander then $H = (V, E_H)$ turns out to be also an expander, w.h.p. The proof proceeds by showing that the probability that RAES completes in T rounds and H is not an ε-expander is $\mathcal{O}(n^{-\gamma})$ for a constant $\gamma > 0$. Combined with Theorem 4, this proves Theorem 5, with $T = \mathcal{O}(\log n)$.

In the next subsection we sketch the main arguments we use in the proof. Then in the successive subsections we provide the detailed proof.

4.1 Overview of the proof.

The probability distribution of the links yielded by RAES, and the ways in which the links are correlated, are very difficult to analyze. In order to cope with such technical issue, we prove Theorem 5 by using a compression argument: we show that the random choices of the algorithm that lead to a non-expading graph can be non-trivially compressed, and hence have low probability.

We think of each node as having access to a sequence of $Td \log \Delta$ random bits and the protocol as being deterministic as a function of these n local sequences of random bits (see Fig. 1). We will show that any sequence of $nTd \log \Delta$ bits leading the protocol to stabilize within T rounds to a non-expanding graph can be losslessly described using $nTd \log \Delta - \Omega(\log n)$ bits. This will prove that the protocol stabilizes to an expanding graph with high probability.

Let $R \in \{0,1\}^{nTd \log \Delta}$ be a bit string that leads to a non-expanding set, i.e., a set S, with size $|S| = s \leqslant n/2$, having at most $\varepsilon |S| d$ outgoing links in H. The compression of such a bit string R is based on two main ideas. Since the number of links in E_H with both endpoints inside S is large, the first main idea is to use less than $\log \Delta$ bits to encode the destination of each accepted requests originated from nodes in S whenever this destination belongs to S. The second main idea is to encode the destinations of rejected requests with less than $\log \Delta$ bits. Indeed, roughly speaking, for each link request that gets rejected at some round, there are at least further cd link requests (in the current or previous rounds) towards the same "bad" destination. Since there are a total of dn requests that need to be accepted and a total of cdn available accepting slots, the number of such "bad" destinations needs to be small, thus the destinations of rejected requests may be compressed.

For clarity sake, we think of the original available randomness $R \in \{0,1\}^{nTd \log \Delta}$ organized as an $n \times Td$ matrix $\mathcal{M}$, where each entry is a block of $\log \Delta$ bits (see Fig. 1). The compressed counterpart of $\mathcal{M}$, denoted as $\mathcal{C}$ (see Fig. 2), consists of three tables (a detailed description of each table can be found in the next subsection).

In order to implement the first idea, for each node $v \in S$ we need to identify, in its row of the uncompressed representation $\mathcal{M}$, which slots of $\log \Delta$ bits refer

to destinations in S of accepted link requests. Notice that this cannot be done naively, indeed even just identifying the set of slots of accepted requests would naively require $\log \binom{Td}{d}$ bits. However, we can first encode the number ℓ_v of used slots, with a prefix free encoding[8] requiring at most $\log \ell_v$ bits and then the set of d slots referring to accepted requests by using $\log \binom{\ell_v}{d}$ bits (see Fig. 2: Field 1 in Table 2). While for some "unlucky" nodes ℓ_v can be large, the overall amortized number of bits $\sum_{v \in S} \left[\mathcal{O}(\log \ell_v) + \log \binom{\ell_v}{d} \right]$ turns out smaller than $s \log \binom{Td}{d}$. Once we have identified the set of accepted requests, we can identify the set of those referring to destinations in S (see Fig. 2: Field 2 in Table 2) and, finally, we can encode each of those destinations by using $\log[(1-\delta_v)\Delta]$ bits instead of $\log \Delta$ bits, where δ_v is the fraction of neighbors of v outside S (see Fig. 2: Field 3 in Table 2). Notice that we can identify which requests of each node end up inside and outside S by encoding the set S itself, once and for all, using $\mathcal{O}(\log s) + \log \binom{n}{s}$ bits (see Fig. 2: Table 1).

In order to implement the second main idea, for each node $v \in S$ we need to identify the destinations of its $\ell_v - d$ rejected link requests. Notice that each rejected request ends up on a node, say w, receiving at least further cd requests. Those further requests include requests *accepted* by w in some previous round and requests *rejected* by w in the current round. We exploit that property to reduce the number of bits used to encode such destinations: roughly speaking, at each round t we distinguish between *semi-saturated* and *critical* nodes. We call semi-saturated at round t a node that already accepted at least $cd/2$ requests up to round $t-1$. Notice that (i) the number of semi-saturated nodes can never exceed $2n/c$ and (ii) we already know the set of semi-saturated nodes at round t, if we know the accepted requests of all nodes up to round $t-1$. Hence, we can encode each request to a semi-saturated node by using only $\log(2n/c)$ bit (notice that this is smaller than $\log \Delta$ whenever $c > 2/\alpha$). In order to distinguish which ones of the $\ell_v - d$ rejected destinations refer to semi-saturated nodes and which ones refer to critical nodes we use further $\ell_v - d$ bits. Finally, for critical nodes (i.e., destinations of rejected requests that are not semi-saturated) we first encode once and for all the set of such nodes at each round, using $\mathcal{O}(\log c_t) + \log \binom{n}{c_t}$ for each round t, so that we can encode the destination of a rejected request toward a critical node at round t using only $\log c_t$ bits.

Summing up all the contributions involved (see Section 4.4 for all the details) we end up encoding a string $R \in \{0,1\}^{nTd \log \Delta}$ leading to a non-expanding set with a bit string of length $nTd \log \Delta - \Omega(\log n)$. Thus the overall number of bit strings leading to non-expanding sets is at most an $\mathcal{O}(n^{-c})$ fraction of all the bit strings, for some $c > 0$.

4.2 The compressed representation: Full description.

We use the following notation throughout the remainder of the paper. For a node $v \in S$, we denote by δ_v the fraction of v's edges in E that have an end-point

[8] See, e.g., *Elias δ-coding* (https://en.wikipedia.org/wiki/Elias_delta_coding).

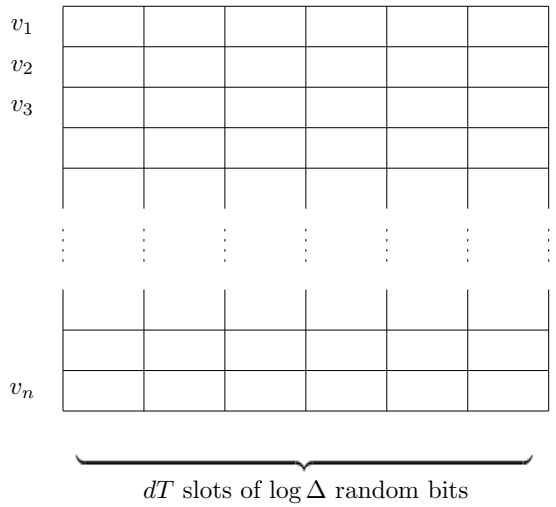

Figure 1: Uncompressed representation $\mathcal{M}$ of R

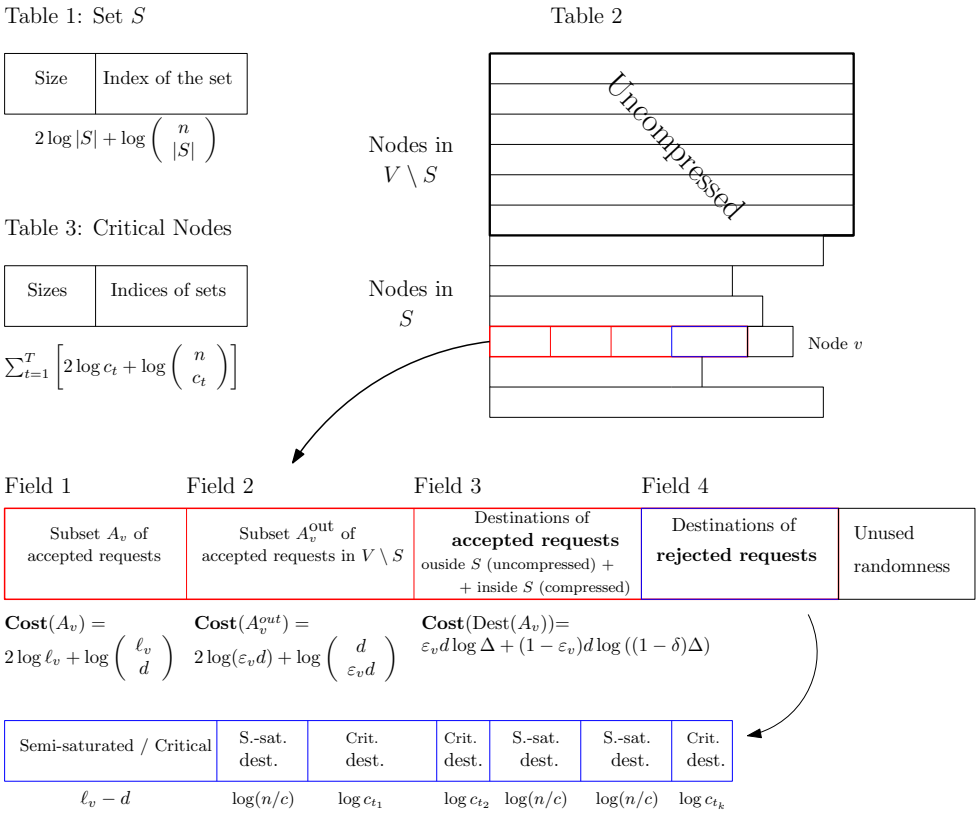

Figure 2: Compressed representation $\mathcal{C}$ of R

in $V - S$, i.e.,

$$\delta_v \cdot \Delta \;=\; e_G(v, V - S) \;\text{ and }\; \delta \;=\; \frac{1}{s} \sum_{v \in S} \delta_v \,. \tag{4}$$

We also denote by ε_v the fraction of v's accepted link requests (so edges of subgraph H) with end-points in $V - S$, i.e.,

$$\varepsilon_v \cdot d \;=\; e_H(v, V - S) \;\text{ and }\; \varepsilon = \frac{1}{s} \sum_{v \in S} \varepsilon_v \,. \tag{5}$$

In the paragraphs that follow, we describe how the evolution of the protocol is encoded in the presence of a non-expanding subset S (with $|S| = s \leqslant n/2$ without loss of generality).

In the remainder, we repeatedly use the following facts:

- **Node numbering:** when representing destinations of link requests submitted by nodes of the network, we can use the fact that the encoding and decoding algorithms have full knowledge of the underlying graph. In particular, we assume a total ordering of the nodes is defined, so that a node u is simply specified by an integer in $\{1, ..., n\}$, denoting u's position in this ordering. At the same time, we can use a local numbering to represent the neighbors of a given node v. For example, if v's neighbors are the nodes $\{2, 5, 8\}$ with respect to the global ordering, node 5 can be represented as 2 with respect to v, i.e., the second neighbor of v with respect to the global ordering.

- **Subset encoding:** given the set $[k]$ of the first k integers, we represent any subset $S \subset [n]$ by its position i in the lexicographic order of all subsets of $[n]$ of size $|S|$. In order to completely specify S, we separately encode its size in a prefix-free way using $2 \log |S|$ bits, and its position i in the lexicographic order using $\log \binom{n}{|S|}$ bits.

We next discuss the compressed encoding we use. We remark that, as argued in [19, Section 7], we can avoid taking ceilings in the expressions which measure the number of bits necessary for the encoding.

Unused randomness.

For every node v, we have enough randomness to describe exactly dT choices. If v completes its execution of the protocol after performing ℓ_v requests, the remaining randomness (corresponding to $dT - \ell_v$ requests) is not used. This unused randomness is both present in the uncompressed representation $\mathcal{M}$ and in its compressed counterpart $\mathcal{C}$ and is represented as is, thus corresponding to $\sum_{v \in V} (dT - \ell_v) \log \Delta$ bits in both $\mathcal{M}$ and $\mathcal{C}$.

Table 1: The set S.

We represent S in $\mathcal{C}$ by writing the number $s := |S|$ in a prefix-free way using $2\log s$ bits, and then writing the number k such that S is the k-th set of size s in lexicographic order, which takes $\log \binom{n}{s}$ bits. Using prefix δ-codes, in total, the cost to encode S is

$$\mathbf{Cost}(S) = 2\log s + \log \binom{n}{s}. \tag{6}$$

Table 2, upper part: Randomness of nodes in $V - S$.

We represent the randomness of nodes in $V - S$ as it is, with no gain or loss. Notice that, thanks to Table 1, a decoder can infer that the first $(n - s)$ rows of Table 2 (see Fig 2) describe the executions of every node in $V - S$. This is a fixed-length encoding formed by $(n - s)$ rows, each consisting of dT blocks of $\log \Delta$ bits: hence a decoder knows where the lower portion of Table 2 encoding the executions of nodes in S begins.

For every node $v \in S$, the lower part of Table 2 contains a *variable-length* row, in turn consisting of a set of consecutive *fields*, which encode the following information.

Table 2, Field 1: Subset A_v of requested links originating from v that are accepted.

This field consists of two parts. In the first part we write, in a prefix free way, the number ℓ_v, using $2\log \ell_v$ bits. As a second part of this field, we specify the subset of the d accepted link requests among the ℓ_v submitted by v.[9] To this purpose, we again encode the integer i, such that the d accepted requests correspond to the i-th subset of $\{1, \ldots, \ell_v\}$ of size d, in lexicographic order. The overall cost incurred for this field is thus

$$\mathbf{Cost}(A_S) = \sum_{v \in S} \mathbf{Cost}(A_v) = \sum_{v \in S} 2\log \ell_v + \log \binom{\ell_v}{d}. \tag{7}$$

Remark: Note that this field allows to iteratively infer the round in which each request was submitted by v. Also notice that the subset of rejected link requests originating from v can be derived as the complement of subset A_v.

Table 2, Field 2: Subset $A_v^{\mathbf{out}} \subseteq A_v$ of accepted links originating from v to $V - S$.

For a node $v \in S$, recall that $\varepsilon_v d$ is the number of outgoing accepted links from v into $V - S$. In this field, we encode the subset A_v^{out} of such accepted links, using the same encoding used for subset A_v in the first field. We can thus

[9] Recall that we are encoding executions of the algorithm that terminate within T rounds.

recover the relative positions of such accepted requests in the overall sequence of the ℓ_v requests made by v. In total, this cost is

$$\begin{aligned}\mathbf{Cost}(A_S^{\text{out}}) &= \sum_{v\in S} \mathbf{Cost}(A_v^{\text{out}}) \\ &= \sum_{v\in S} 2\log(\varepsilon_v d) + \log\binom{d}{\varepsilon_v d}. \end{aligned} \tag{8}$$

Remark: observe that the encoding of A_v^{out} is relative to subset A_v. For example, if $d = 4$ and $A_v = \{1, 3, 5, 6\}$, we would know that the first, third, fifth and sixth requests placed by v were accepted. Moreover, if $A_v^{\text{out}} = \{2, 3\}$, we would know that out of these, the second and third (i.e., the third and fifth request out of the ℓ_v submitted by v) had destination in $V - S$.

Note that the boundary between the first field and the second one above is uniquely determined by the value of ℓ_v, which is encoded in a prefix-free way. The same holds for the second field.

Table 2, Field 3: Destinations of accepted links originating from v.

This field consists of two parts. In the first part, we represent accepted links with destinations in $V - S$ as they are (i.e., using $\log \Delta$ bits), with no gain or loss. In the second part, we represent destinations of accepted links in S using $\log((1 - \delta_v)\Delta)$ bits instead of $\log \Delta$. Overall, the cost we incur is

$$\mathbf{Cost}(\text{Dest}(A_S)) = \sum_{v\in S} (1 - \varepsilon_v) d \log((1 - \delta_v)\Delta) + \varepsilon_v d \log \Delta\,. \tag{9}$$

Remark. Note that here we are using a local numbering for neighbors of v that belong to S. Moreover, thanks to the information encoded in the previous fields (i.e. the size of A_v^{out} and that of A_v) we can use a standard block code for both the above parts since we know exactly their respective lengths.

Table 2, Field 4: Destinations of rejected requests originating from v.

We finally compress the encoding of the destinations of rejected requests. In order to do so, we first introduce the following notions.

Definition 8 (Semi-saturated and Critical)**.** *We call a node w*

- semi-saturated at round t*, if the number of accepted incoming links up to round $t-1$, plus the number of requested links at round t originating from nodes in $V - S$ is at least $cd/2$.*
- critical at round t*, if it is* not *semi-saturated at round t but it has more than cd links (accepted or requested) at round t (note that this implies that w received more than $cd/2$ requests from S at round t).*

We will make use of the following facts.

Lemma 9. *For every round t, it holds that:*

- *The number of semi-saturated nodes is at most $2n/c$.*
- *The number of critical nodes is at most n/c.*

Proof. Consider a node that is semi-saturated at round t. This node was the recipient of at least $cd/2$ link requests, that it either accepted *before* round t, or it received in round t. Since, from the definition of RAES, for every node, the overall number of its link requests that are accepted within round $t-1$, plus the number of link requests it issues at round t cannot exceed d, we have a total of at most dn such requests over the entire network. This immediately implies that the number of semi-saturated nodes at round t cannot exceed $2n/c$. The argument for the number of critical nodes at round t proceeds along the same lines and is omitted for the sake of brevity. □

In what follows we represent the subsets of semi-saturated and critical nodes.

- **The subset of semi-saturated nodes at each step.** From its definition, the set of semi-saturated nodes needs not be represented explicitely. In fact, for every round t, this set is uniquely determined by the evolution of the protocol (and thus by the corresponding portions of our Tables) up to round $t-1$ and by link requests issued by nodes in $V-S$ at round t, whose randomness is represented as it is (see Table 2, upper portion).
- **Table 3: The subset of critical nodes at each step.** We represent the subsets of critical nodes in each round explicitely. Let C_t be the set of critical nodes at round t and let $c_t := |C_t|$. We represent all such sets in a separate table (see Table 3 in Figure 2). This table consists of two fields. The first is the sequence of the critical set sizes, encoded in a prefix-free way. The second field is the sequence of the integers representing $C_t \subset V$, for $t = 1, \ldots, T$. Note that, the length of the field encoding C_t is completely determined once we know c_t. Overall, encoding information in Table 3 has cost

$$\mathbf{Cost}(C) = \sum_{t=1}^{T} 2\log c_t + \log \binom{n}{c_t}. \tag{10}$$

Given this premise, this field consist of two parts. The first part is a sequence of exactly $\ell_v - d$ bits. The i-th such bit specifies whether the destination of the i-th rejected request was a semi-saturated or critical node in the round in which the request was issued. The second part of the field is simply the sequence of destinations of rejected requests, encoded in compressed form thanks to Lemma 9. Specifically, for each round t, we represent each rejected connection toward to a critical node using $\log c_t$ bits (recall that we explicitely represent C_t), and each other rejected connection, which necessarily goes to a semi-saturated node, using $\log(2n/c)$ bits.

To compute the corresponding cost of representing destinations of rejected requests, let $rc_t(v)$ be the number of rejected requests from v to critical nodes in round t, and let $rss(v)$ be the overall number of rejected connection requests from v to semi-saturated nodes, over the entire process. Then, the overall cost of encoding the destinations of rejected requests from v is

$$\begin{aligned}\mathbf{Cost}(\mathrm{Dest}(\mathrm{Rej})) &= (\ell_v - d) + rss(v) \cdot \log \frac{2n}{c} \\ &\quad + \sum_{t=1}^{T} rc_t(v) \cdot \log c_t \ . \end{aligned} \tag{11}$$

Observe that the additive term $(\ell_v - d)$ in the equation above corresponds to the aforementioned first part of the field.

4.3 Decoding algorithm.

We show correctness of our encoding, discussing how the entire evolution of the protocol can be recovered from its compressed encoding without loss of information. Before describing this decoding algorithm, it is useful to define, for the remainder of this section, the notion of *state of* RAES*'s execution* at round t.

Definition 10. *The state* $\mathbf{X}_t$ *of* RAES*'s execution at time* t *is a vector, whose component* $\mathbf{X}_t(v)$ *is the ordered sequence of the destinations of all link requests issued by* v *in round* t*.*

We note that knowledge of $\{\mathbf{X}_1, \ldots, \mathbf{X}_t\}$ allows to fully characterize the evolution of the process up to round t. In particular, for every round $i = 1, \ldots, t$, we can tell exactly which requests were accepted and which were rejected in that round.

Further notation used in this subsection.

For a node v and a round t, we define by $x_t(v)$ and $a_t(v)$ respectively the overall number of link requests submitted by v in round t and the number of those that were accepted. We let $x_{\leqslant t}(v) = \sum_{i=1}^{t} x_i(v)$ and $a_{\leqslant t}(v) = \sum_{i=1}^{t} a_i(v)$ for conciseness (note that $a_t(v) \leqslant x_t(v)$ and $a_{\leqslant t}(v) \leqslant x_{\leqslant t}(v)$ by definition). For every $v \in V$, we denote by $\mathrm{Dest}_t(v)$ the set of destinations of requests issued by v in round t. We denote by SS_t and C_t respectively the subsets of semi-saturated and critical nodes in round t.

We next outline the main steps of a decoding algorithm $Dec(G, \mathcal{C})$. The algorithm takes as input the underlying graph G and the compressed encoding $\mathcal{C}$ and it returns the evolution of RAES over the at most T steps of its execution. More precisely, for every t, $Dec(G, \mathcal{C})$ returns a special symbol $\emptyset$ if RAES completed its execution before time t. Otherwise, $Dec(G, \mathcal{C})$ returns $\mathbf{X}_t$, i.e., for every v, the sequence of requests issued by v in round t. Note that this is enough to recover $\mathcal{M}$, since unused randomness is represented as is both in $\mathcal{M}$

and $\mathcal{C}$. In particular we show how, given G, $\mathcal{C}$ and $\{\mathbf{X}_1, \ldots, \mathbf{X}_{t-1}\}$, it is possible to recover $\mathbf{X}_t$.[10] The main steps of the algorithm are summarized as Algorithm 2 below, while details on how each piece of information can be recovered from $\mathcal{C}$ have been discussed in Section 4.2. This is enough to prove that the compressed encoding is lossless.

Algorithm 2 DEC(G, $\mathcal{C}$)

Identify S from Table 1
for $t = 1, \ldots, T$ **do**
    Use $\{\mathbf{X}_1, \ldots, \mathbf{X}_{t-1}\}$ to compute $x_{\leqslant t-1}(v)$ and $a_{\leqslant t-1}(v)$, for every $v \in V$
    if $a_{\leqslant t-1}(v) = d$ for every $v \in V$ **then** return $\emptyset$
    end if
    for $v \in V - S$ **do**
        Look up v's row in Table 2, using $x_{\leqslant t-1}(v)$ and $a_{\leqslant t-1}(v)$ to identify the set $\mathrm{Dest}_t(v)$ of the destinations of the $d - a_{\leqslant t-1}(v)$ requests that were submitted by v in round t
        Use $\{\mathbf{X}_1, \ldots, \mathbf{X}_{t-1}\}$ and $\mathrm{Dest}_t(V - S)$ (the latter computed in the previous step) to identify the subset SS_t of semi-saturated nodes in round t
        Use Table 3 to identify the subset C_t of critical nodes in round t
    end for
    for $v \in S$ **do**
        Use Field 1 of v's row in $\mathcal{C}$ to identify the subset of v's accepted requests that were submitted in round t and compute their number $a_t(v)$
        Use information collected in the previous step, Field 2 and Field 3 to identify the destinations of accepted requests submitted by v in round t
        Use Field 4 and SS_t and C_t computed above to identify the destinations of rejected requests submitted in round t
    end for
end for
return $\mathbf{X}_t$

4.4 Rate of compression.

In this subsection, we show that, if R represents an execution terminating and returning a *non expanding* graph H, the corresponding encoding according to the scheme presented in the previous section uses $ndT \log \Delta - \Omega(\log n)$. In more detail, we apply our encoding scheme described in the previous subsection to any subset $S \subset V$ that is not an "ε-expander" in the graph H returned by RAES.

The analysis of the achieved compression rate proceeds by carefully bounding the costs of the compressed representation R' and comparing them with their counterparts in the uncompressed representation R. We first show that the additive costs (with respect to R) of representing the non-expanding set S (see Table 1 of Fig. 2 and (6)) and the subsets A_S^{out} of accepted requests with

[10] Note that $\mathbf{X}_0$ simply contains an empty request sequence for every v.

destinations in $V-S$ (see Field 2 in Table 2 and (8)) are more than compensated by the compression achieved in the representation of accepted requests with destinations in S (see Field 3 in Table 2 and (9)), with total savings $\Omega(ds \log \frac{n}{s})$. This first step corresponds to bounding the partial cost $\mathbf{Cost}(S)+\mathbf{Cost}(A_S^{\text{out}})+\mathbf{Cost}(\text{Dest}(A_S))$ and it is provided in Lemma 12.

If this is intuitively the key argument, it is neglecting the fact that we now need to identify the subset A_S of requests originating from S that are accepted (see Field 1 of Table 2 and (7)). For each node, this cost depends on the number of failures and in general cannot be compensated by the aforementioned savings. To this purpose, we need to exploit the further property that, for each node, failures have destinations that, for each round t, correspond to an $\mathcal{O}(1/c)$ fraction of the vertices. Thanks to the use of semisatured and critical nodes (see Field 4 of Table 2), compressing the destinations of these requests allows to compensate the aforementioned cost almost entirely. This second step corresponds to bounding the partial cost $\mathbf{Cost}(A_S) + \mathbf{Cost}(C) + \mathbf{Cost}(\text{Dest}(\text{Rej}))$ and it is provided in Lemma 14.

We state and prove a useful bound, that easily follows from the expansion property of the underlying graph G and Lemma 2.

Lemma 11. *Let $G = (V, E)$ be a Δ-regular graph and let λ be the second largest eigenvalue of G's adjacency matrix. Then, for any subset $S \subseteq V$, it holds that*

$$1 - \delta \leqslant \frac{s}{n} + \frac{\lambda}{\Delta}\,.$$

where $s = |S|$ and δ is defined as in (4).

Proof. From the definition of δ_v we have that the number of edges with both end-points in S is

$$e(S, S) = \sum_{v \in S}(1 - \delta_v)\Delta = (1 - \delta)s\Delta\,.$$

From Lemma 2 it thus follows that

$$(1 - \delta) = \frac{e(S, S)}{s\Delta} \leqslant \frac{1}{2s\Delta}\left(\frac{\Delta s^2}{n} + \lambda s\right) = \frac{1}{2}\left(\frac{s}{n} + \frac{\lambda}{\Delta}\right)\,.$$

□

As a first, crucial step of our compression analysis, we evaluate the cost $\mathbf{Cost}(\text{Dest}(A_S))$ - see (9) - of representing the destinations corresponding to the subset A_S of accepted link requests from nodes in S. We decided to isolate this step since it is the only one in which we make use of the expansion property of the underlying graph G, stated in the lemma above.

Lemma 12. *Under the hypotheses of Theorem 5, the cost $\mathbf{Cost}(\text{Dest}(A_S))$ - see (9) - of representing the destinations corresponding to the subset A_S of accepted link requests from nodes in S satisfies the following bound:*

$$\mathbf{Cost}(\text{Dest}(A_S)) \leqslant \sum_{v \in S}(1 - \varepsilon_v)d \log((1 - \delta_v)\Delta) +$$

$$+\sum_{v\in S}\varepsilon_v d\log\Delta \leqslant sd\log\Delta - \frac{1-\varepsilon}{2}sd\log\frac{n}{s} + 2\varepsilon ds\,. \tag{12}$$

Proof. We directly give a lower bound to the savings achieved with respect to the cost of the uncompressed representation, the latter being $sd\log\Delta$. Namely, we prove that

$$\begin{aligned} sd\log\Delta - \left(\sum_{v\in S}(1-\varepsilon_v)d\log((1-\delta_v)\Delta) + \sum_{v\in S}\varepsilon_v d\log\Delta\right) \\ \geqslant \frac{1-\varepsilon}{2}ds\log\frac{n}{s} - 2\varepsilon ds\,, \end{aligned} \tag{13}$$

whence (12) immediately follows. First of all, the LHS of (13) can be written as

$$\begin{aligned} &\sum_{v\in S} d\log\Delta - \sum_{v\in S}\varepsilon_v d\log\Delta + \\ &- \left(\sum_{v\in S}(1-\varepsilon_v)d\cdot\log((1-\delta_v)\Delta)\right) \\ &= d\sum_{v\in S}(1-\varepsilon_v)\log\Delta - d\sum_{v\in S}(1-\varepsilon_v)\log((1-\delta_v)\Delta) \\ &= d\sum_{v\in S}(1-\varepsilon_v)\log\frac{1}{1-\delta_v}\,. \end{aligned} \tag{14}$$

Next, we consider two cases.

Case $s < \alpha\Delta$. By definition of δ_v, we easily get that

$$(1-\delta_v)\Delta = e_G(v,S) \leqslant s$$

that immediately implies

$$1-\delta_v \leqslant \frac{s}{\Delta}\,. \tag{15}$$

From (14), we get

$$\begin{aligned} d\sum_{v\in S}(1-\varepsilon_v)\log\frac{1}{1-\delta_v} &\geqslant d\sum_{v\in S}(1-\varepsilon_v)\log\frac{\Delta}{s} \\ = (1-\varepsilon)sd\log\frac{\Delta}{s} &> \frac{1-\varepsilon}{2}sd\log\frac{n}{s}\,, \end{aligned}$$

where we used (15) to write the first inequality, while the last inequality follows from the definition $\varepsilon = (1/s)\sum_{v\in S}\varepsilon_v$ and since, in this case,

$$\frac{\Delta}{s} > \frac{1}{\alpha} = \frac{n}{\Delta}\,,$$

which in turn implies

$$\frac{\Delta}{s}\cdot\frac{\Delta}{s}=\left(\frac{\Delta}{s}\right)^2>\frac{n}{\Delta}\cdot\frac{\Delta}{s}=\frac{n}{s}\,.$$

Case $\alpha\Delta\leqslant s\leqslant n/2$. In this case, the proof is a bit more articulated. To begin, we can write

$$\begin{aligned}
&\sum_{v\in S}(1-\varepsilon_v)\log\frac{1}{1-\delta_v}=-\sum_{v\in S}(1-\varepsilon_v)\log(1-\delta_v)\\
&=-(1-\varepsilon)s\sum_{v\in S}\frac{1-\varepsilon_v}{(1-\varepsilon)s}\log(1-\delta_v)\\
&\geqslant-(1-\varepsilon)s\log\frac{\sum_{v\in S}(1-\varepsilon_v)(1-\delta_v)}{(1-\varepsilon)s}\\
&\geqslant-(1-\varepsilon)s\log\frac{\sum_{v\in S}(1-\delta_v)}{(1-\varepsilon)s}=(1-\varepsilon)s\log\frac{1-\varepsilon}{1-\delta}\,. \qquad (16)
\end{aligned}$$

Here, to derive the third inequality we used Jensen's inequality on the function

$$\sum_{v\in S}\frac{1-\varepsilon_v}{(1-\varepsilon)s}\log(1-\delta_v)\,,$$

which is a convex combination of values of a concave function.

Next, going back to (14), from (16) and Lemma 11 we easily get

$$\begin{aligned}
&d\sum_{v\in S}(1-\varepsilon_v)\log\frac{1}{1-\delta_v}\geqslant(1-\varepsilon)ds\log\frac{1-\varepsilon}{1-\delta}\\
&\geqslant(1-\varepsilon)ds\log\frac{(1-\varepsilon)n\Delta}{\lambda n+\Delta s}\,. \qquad (17)
\end{aligned}$$

Since we are in case $s\geqslant\alpha\Delta$ and, by hypothesis of Theorem 5, $\lambda\leqslant\varepsilon\alpha^2\Delta$ and $\Delta=\alpha n$, it holds that $\lambda\leqslant\varepsilon\alpha s=\varepsilon s\Delta/n$. Hence $s\Delta+\lambda n\leqslant(1+\varepsilon)s\Delta$ and it follows that

$$\frac{1}{s}\leqslant(1+\varepsilon)\frac{\Delta}{s\Delta+\lambda n}\,.$$

This, in turn, implies that

$$\frac{(1-\varepsilon)n\Delta}{\lambda n+\Delta s}\geqslant\frac{n}{s}\cdot\frac{1-\varepsilon}{1+\varepsilon}\,.$$

From the above inequality and (17), we easily get

$$\begin{aligned}
&d\sum_{v\in S}(1-\varepsilon_v)\log\frac{1}{1-\delta_v}\geqslant(1-\varepsilon)ds\log\left(\frac{n}{s}\cdot\frac{1-\varepsilon}{1+\varepsilon}\right)\\
&\geqslant(1-\varepsilon)ds\log\frac{n}{s}-(1-\varepsilon)ds\log\left(1+\frac{2\varepsilon}{1+\varepsilon}\right)
\end{aligned}$$

$$\geqslant (1-\varepsilon)ds\log\frac{n}{s} - (1-\varepsilon)ds\left(\frac{2\varepsilon}{1+\varepsilon}\right)$$
$$\geqslant (1-\varepsilon)ds\log\frac{n}{s} - ds2\varepsilon\,.$$

□

The above lemma is then used below to bound the first partial cost of our compressed representation of R, i.e., $\mathbf{Cost}(S)+\mathbf{Cost}(A_S^{\text{out}})+\mathbf{Cost}(\text{Dest}(A_S))$.

Lemma 13. *Under the hypotheses of Theorem 5, the overall cost of representing sets S, A_S^{out} and $\text{Dest}(A_S)$ - see (6), (8), and (9) - satisfies the following bound*

$$\begin{aligned}&\mathbf{Cost}(S)+\mathbf{Cost}(A_S^{out})+\mathbf{Cost}(\text{Dest}(A_S)) \leqslant\\ &\leqslant 3s\log\frac{n}{s}+ds\log\Delta-\frac{1-13\varepsilon\log(1/\varepsilon)}{2}ds\log\frac{n}{s}+2\varepsilon ds\end{aligned}\tag{18}$$

Proof. As for $\mathbf{Cost}(S)$, from (6), notice that

$$2\log s+\log\binom{n}{s}\leqslant 3s\log\frac{n}{s}\,,\tag{19}$$

since we are assuming $s\leqslant n/2$ and it holds $\log\binom{n}{s}\leqslant s\log(ne/s)$.

As for $\mathbf{Cost}(A_S^{\text{out}})$, the first term in (8) can be bounded as follows

$$\begin{aligned}&2\sum_{v\in S}\log(\varepsilon_v d)=2\log\prod_{v\in S}(\varepsilon_v d)\\ &\leqslant 2\log\left(\frac{\sum_{v\in S}\varepsilon_v d}{s}\right)^s=2s\log(\varepsilon d)\,,\end{aligned}\tag{20}$$

where the second inequality follows from the AM-GM inequality [23], while the last equality follows from the definition of ε (seei (5)). Moreover, the second term in (8) can be bounded as follows

$$\begin{aligned}&\sum_{v\in S}\log\binom{d}{\varepsilon_v d}\leqslant d\sum_{v\in S}\varepsilon_v\log\frac{e}{\varepsilon_v}\\ &=\varepsilon ds\log e+d\sum_{v\in S}\varepsilon_v\log\frac{1}{\varepsilon_v}\leqslant\varepsilon ds\log e+\varepsilon ds\log\frac{1}{\varepsilon}\,.\end{aligned}\tag{21}$$

Here, the second equality follows again from the definition of ε, while the third inequality follows since the optimum of the following problem

$$\max g(x_1,...,x_k)=\sum_{i=1}^k x_i\log\frac{1}{x_i}$$
$$\sum_{i=1}^k x_i=B\quad\text{with } x_i\geqslant 0\,,\ i=1,\ldots,k$$

is achieved when $x_1 = \ldots = x_k = B/k$.

Finally, combining (19), (20), (21), and (12) given in Lemma 12, we get

$$\begin{aligned}
&\mathbf{Cost}(S) + \mathbf{Cost}(A_S^{\text{out}}) + \mathbf{Cost}(\text{Dest}(A_S)) \\
&\leqslant 3s \log \frac{n}{s} + \underbrace{2s \log(\varepsilon d) + \varepsilon ds \log e + \varepsilon ds \log \frac{1}{\varepsilon}}_{*} + \\
&+ ds \log \Delta - \frac{1-\varepsilon}{2} ds \log \frac{n}{s} + 2\varepsilon ds \\
&\leqslant 3s \log \frac{n}{s} + ds \log \Delta - \frac{1 - 13\varepsilon \log(1/\varepsilon)}{2} ds \log \frac{n}{s} + 2\varepsilon ds \,, \qquad (22)
\end{aligned}$$

where the last inequality holds, since each of the starred terms is at most $2\varepsilon \log(1/\varepsilon) ds \log \frac{n}{s}$, whenever $s \leqslant n/2$, and $\varepsilon \leqslant 1/2$. □

In the next lemma, we bound the remaining part of our compressed representation of R, i.e., the costs $\mathbf{Cost}(A_S) + \mathbf{Cost}(C) + \mathbf{Cost}(\text{Dest}(\text{Rej}))$.

Lemma 14. *Under the hypotheses of Theorem 5, the overall cost of representing sets A_S, $\mathbf{Cost}(C)$, and* Dest(Rej)) *- see (7), (10), and (11) - satisfies the following bound*

$$\begin{aligned}
&\mathbf{Cost}(A_S) + \mathbf{Cost}(C) + \mathbf{Cost}(\text{Dest}(\text{Rej})) \\
&\leqslant \log \Delta \sum_{v \in S} (\ell_v - d) + \frac{1}{4} ds \,.
\end{aligned}$$

Proof. As for $\mathbf{Cost}(A_S)$, from (7), we next show that

$$\begin{aligned}
\mathbf{Cost}(A_S) &= \sum_{v \in S} \left(2 \log \ell_v + \log \binom{\ell_v}{d} \right) \\
&\leqslant \sum_{v \in S} \left(5d(\ell_v - d) + \frac{d}{4} \right) . \qquad (23)
\end{aligned}$$

When $\ell_v = d$, the last inequality follows from the hypothesis $d \geqslant 44$, since $2 \log \ell_v + \log \binom{\ell_v}{d} = 2 \log d \leqslant \frac{d}{4}$. When $\ell_v > d$, we note that

$$5d \geqslant \underbrace{d(\log e + 1)}_{(\star)} + \underbrace{2d}_{(\star\star)},$$

and we observe that $(\star\star)$ accounts for the first term of $\mathbf{Cost}(A_S)$

$$2d(\ell_v - d) \geqslant 2 \log \ell_v,$$

and that $(\star)$ accounts for the second term of $\mathbf{Cost}(A_S)$

$$\log \binom{\ell_v}{d} \leqslant d \log \frac{e \ell_v}{d} = d \log e + d(\log \ell_v - \log d)$$

$$\leqslant d \log e + d(\ell_v - d) = d(\log e + 1)(\ell_v - d).$$

As for $\mathbf{Cost}(C)$, the definition of critical node at round t implies that each critical node at round t is receiving more than $cd/2$ of its incoming requests from nodes in S. As a consequence, we get

$$\sum_{v \in S} rc_t(v) > \frac{cd}{2} c_t \,,$$

where we recall that $rc_t(v)$ is the number of rejected requests from node v to critical nodes and c_t is the size of the subset of nodes which turn to be critical at round t. For the first term in (10) this implies

$$\begin{aligned} 2\sum_{t=1}^{T} \log c_t &\leqslant 2 \sum_{t=1}^{T} \log \left(\frac{2}{cd} \sum_{v \in S} rc_t(v) \right) \\ &< \frac{4}{cd} \sum_{t=1}^{T} \sum_{v \in S} rc_t(v) \leqslant \frac{4}{cd} \sum_{v \in S} (\ell_v - d) \,, \end{aligned} \tag{24}$$

where to derive the last inequality we exchanged the order of summation and used the fact that $\sum_{t=1}^{T} rc_t(v) \leqslant \ell_v - d$. As for the second term in (10), we get

$$\begin{aligned} \log \binom{n}{c_t} &\leqslant c_t \log \frac{en}{c_t} \\ &\leqslant \frac{2}{cd} \sum_{v \in S} rc_t(v) \log \frac{en}{c_t} \leqslant \sum_{v \in S} rc_t(v) \log \frac{2n}{c \cdot c_t} \,. \end{aligned} \tag{25}$$

Here, to derive the third inequality we used the following

Claim 15. *If c is large enough that $d \geqslant \frac{2}{c} \log \frac{ce}{2}$, then*

$$\log \left(\frac{en}{c_t} \right)^{\frac{2}{cd}} \leqslant \log \frac{2n}{c \cdot c_t} \,.$$

The proof of the above inequality follows from simple calculus, recalling that the definition of critical node implies that there are at most n/c of them at any round t.
By combining (23), (24), (25) and (11), we set $\gamma = 1 + 5d + \frac{4}{cd}$ and derive the following upper bound

$$\begin{aligned} &\mathbf{Cost}(A_S) + \mathbf{Cost}(C) + \mathbf{Cost}(\mathrm{Dest}(\mathrm{Rej})) \\ &\leqslant \sum_{v \in S} \left(\gamma(\ell_v - d) + \frac{1}{4} d + \sum_{t=1}^{T} rc_t(v) \log \frac{2n}{c \cdot c_t} + \right. \\ &\left. + rss(v) \cdot \log \frac{2n}{c} + \sum_{t=1}^{T} rc_t(v) \cdot \log c_t \right) \end{aligned}$$

$$
\begin{aligned}
&= \gamma \sum_{v\in S}(\ell_v - d) + \frac{1}{4}ds + \sum_{v\in S}\left(\sum_{t=1}^{T} rc_t(v)\log\frac{2n}{c} + rss(v)\cdot\log\frac{2n}{c}\right)\\
&= \gamma \sum_{v\in S}(\ell_v - d) + \frac{1}{4}ds + \sum_{v\in S}(\ell_v - d)\log\frac{2n}{c}\\
&= \gamma \sum_{v\in S}(\ell_v - d) + \frac{1}{4}ds + \sum_{v\in S}(\ell_v - d)\log\frac{2n}{\sqrt{c}} - \frac{1}{2}\sum_{v\in S}(\ell_v - d)\log c\\
&\overset{(a)}{\leqslant} \gamma \sum_{v\in S}(\ell_v - d) + \frac{1}{4}ds + \sum_{v\in S}(\ell_v - d)\log\Delta - \frac{1}{2}\sum_{v\in S}(\ell_v - d)\log c\\
&\overset{(b)}{\leqslant} \log\Delta \sum_{v\in S}(\ell_v - d) + \frac{1}{4}ds\,, \qquad (26)
\end{aligned}
$$

where we used that we can assume c large enough so that, in (a), $c \geqslant (2/\alpha)^2$ and $\Delta = \alpha n$ imply $\Delta \geqslant 2n/\sqrt{c}$, while in (b) we used $\gamma = 1 + 5d + \frac{4}{cd} \leqslant \frac{1}{2}\log c$. □

Wrap up: Proof of Theorem 5.

From Lemmas 13 and 14 it follows that the total number of bits to encode the executions of nodes in S is as follows (recall that we use $(n-s)dT\log\Delta$ bits for vertices in $V - S$).

$$
\begin{aligned}
&\mathbf{Cost}(S) + \mathbf{Cost}(A_S^{\text{out}}) + \mathbf{Cost}(\text{Dest}(A_S)) +\\
&+\ \mathbf{Cost}(A_S) + \mathbf{Cost}(C) + \mathbf{Cost}(\text{Dest}(\text{Rej}))\\
&\leqslant 3s\log\frac{n}{s} + ds\log\Delta - \frac{1 - 13\varepsilon\log(1/\varepsilon)}{2}ds\log\frac{n}{s} +\\
&+\ 2\varepsilon ds + \log\Delta\sum_{v\in S}(\ell_v - d) + \frac{1}{4}ds\,. \qquad (27)
\end{aligned}
$$

To this cost, we should also add the randomness that was not used, which is exactly $(dT - \ell_v)\log\Delta$ for the generic node v. Our savings are then

$$
\begin{aligned}
\text{Savings} \ &\geqslant\ dsT\log\Delta +\\
&- \left(3s\log\frac{n}{s} + ds\log\Delta - \frac{1 - 13\varepsilon\log(1/\varepsilon)}{2}ds\log\frac{n}{s}\ + \right.\\
&\left. +2\varepsilon ds + \log\Delta\sum_{v\in S}(dT - d) + \frac{1}{4}ds\right)\\
&= -3s\log\frac{n}{s} + \frac{1 - 13\varepsilon\log(1/\varepsilon)}{2}ds\log\frac{n}{s} - \left(\frac{1}{4} + 2\varepsilon\right)ds\,.
\end{aligned}
$$

Finally, the expression above is $\Omega(\log n)$, as soon as ε is a sufficiently small (absolute) constant.

5 Future work

A first interesting open problem is extending the analysis of RAES to non-dense expanders, i.e., to cases in which $\Delta = o(n)$. In this setting, both the proofs of convergence and of expansion might need to be revisited in significant ways. For example, if $\Delta < n/c$, we can no longer guarantee that all nodes will eventually establish d connections: it might well be the case that all neighbours of some node become saturated at some point, before the node itself can see all its requests accommodated. In fact, $\Delta = \Omega(\log n)$ is necessary to ensure that this does not occur with high probability. Another interesting generalization is extending the analysis to the case of non-regular graphs, possibly relying on the corresponding generalization of the Expander Mixing Lemma. Finally, it would be interesting to investigate the robustness of RAES in dynamic settings, in which nodes and/or vertices of the underlying graph G may join or leave the network.

References

[1] Zeyuan Allen-Zhu, Aditya Bhaskara, Silvio Lattanzi, Vahab Mirrokni, and Lorenzo Orecchia. Expanders via local edge flips. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete algorithms (SODA 2016)*, pages 259–269. Society for Industrial and Applied Mathematics, 2016. 5

[2] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012. 2, 5

[3] Petra Berenbrink, Tom Friedetzy, Christiane Lammersen, and Thomas Sauwervald. Parallel randomized load balancing. *Unpublished Manuscript*, 2018. 6

[4] Petra Berenbrink, Kamyar Khodamoradi, Thomas Sauerwald, and Alexandre Stauffer. Balls-into-bins with nearly optimal load distribution. In *Proceedings of the 25th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2013)*, pages 326–335, New York, NY, USA, 2013. ACM. 6

[5] Keren Censor-Hillel, Bernhard Haeupler, Jonathan A. Kelner, and Petar Maymounkov. Global computation in a poorly connected world: fast rumor spreading with no dependence on conductance. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC 2012) New York, NY, USA, May 19 - 22, 2012*, pages 961–970, 2012. 8, 11

[6] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009. 10, 11, 30

[7] Alan M. Frieze and Michael Molloy. Splitting an expander graph. *J. Algorithms*, 33(1):166–172, 1999. 5

[8] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS 2000)*, pages 305–313. IEEE, 2000. 4

[9] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks: Algorithms and evaluation. *Perform. Eval.*, 63(3):241–263, March 2006. 4, 5

[10] Bernhard Haeupler. Simple, fast and deterministic gossip and rumor spreading. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013), New Orleans, Louisiana, USA, January 6-8, 2013*, pages 705–716, 2013. 8, 11

[11] Ioannis Koutis and Shen Chen Xu. Simple parallel and distributed algorithms for spectral graph sparsification. *ACM Transactions on Parallel Computing (TOPC)*, 3(2):14, 2016. 5

[12] Ching Law and K-Y Siu. Distributed construction of random expander networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 2003)*, volume 3, pages 2133–2143. IEEE, 2003. 5

[13] Christoph Lenzen and Roger Wattenhofer. Tight bounds for parallel randomized load balancing: Extended abstract. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC 2011)*, pages 11–20, New York, NY, USA, 2011. ACM. 6

[14] David A. Levin and Yuval Peres. *Markov Chains and Mixing Times: Second Edition*. American Mathematical Society, 2017. 6

[15] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer Science & Business Media, 2013. 3

[16] Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison–Singer problem. *Annals of Mathematics*, pages 327–350, 2015. 2

[17] Colin McDiarmid. Concentration. In *Probabilistic methods for algorithmic discrete mathematics*, pages 195–248. Springer, 1998. 30

[18] Adler Micah, Chakrabarti Soumen, and Rasmussen Lars E. Parallel randomized load balancing. *Random Struct. Algorithms*, 13(2):159–188, 1998. 6

[19] Pat Morin, Wolfgang Mulzer, and Tommy Reddad. Encoding arguments. *ACM Comput. Surv.*, 50(3):46:1–46:36, July 2017. 4, 10, 15, 30, 31

[20] Thomas Moscibroda, Stefan Schmid, and Rogert Wattenhofer. On the topologies formed by selfish peers. In *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing (PODC 2006)*, pages 133–142, New York, NY, USA, 2006. ACM. 4

[21] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. White Paper, 2008. 4

[22] Moni Naor and Udi Wieder. Novel architectures for p2p applications: The continuous-discrete approach. *ACM Trans. Algorithms*, 3(3), August 2007. 5

[23] J. Michael Steele. *The Cauchy-Schwarz master class: an introduction to the art of mathematical inequalities*. Cambridge University Press, 2004. 24

Appendix

A Mathematical Tools

In Section 3, we used the method of bounded differences. In particular, we applied - a slight generalization of[11] - the concentration bound in [6, 17].

Theorem 16. *Let* $\mathbf{Y} = (Y_1, \ldots, Y_m)$ *be independent r.v.s, with* Y_j *taking values in a set* A_j*. Suppose the real-valued funcion* f *defined on* $\prod A_j$ *satisfies*

$$|f(\mathbf{y}) - f(\mathbf{y}')| \leqslant \beta_j$$

whenever vectors $\mathbf{y}$ *and* $\mathbf{y}'$ *differs only in the* j*-th coordinate. Let* μ *be an upper bound to the expected value of r.v.* $f(\mathbf{Y})$*. Then, for any* $M > 0$*, it holds that*

$$\Pr\left(f(\mathbf{Y}) - \mu \geqslant M\right) \leqslant e^{-\frac{2M^2}{\sum_{j=1}^{m} \beta_j^2}}.$$

B Proving Lemma 6 via an encoding argument

We provide here an elegant, alternative proof for the fact RAES on graph G completes its task within a logarithmic number of rounds, w.h.p. The proof relies on a simple encoding argument [19] and it can be seen as a "warm-up" for the much more complex analysis given in Section 4 which makes use of the same approach.

[11] Via a simple coupling argument, in the bound in [6, 17] we can substitute to the expected value an upper bound to it.

Lemma 17. *Let $G = (V, E)$ be any Δ-regular graph with $\Delta = \alpha n$ and $0 < \alpha \leqslant 1$ and let $d \geqslant 1$ be any absolute constant. Then, for any $c > 1/\alpha$, any $\beta > 2$, and any large-enough n, RAES(G, d, c) on graph G terminates within $(\beta/\log(\alpha c))\log n$ rounds with probability at least $1 - n^{-(\beta-2)/2}$.*

Proof. We prove the lemma via an *encoding* argument [19][12]. Notice that any execution of RAES for t rounds is completely determined by a sequence of $tnd\log\Delta$ random bits: Indeed, $\log\Delta$ random bits can be used for each link request of any fixed node, each node makes at most d link requests at each round, and this procedure is repeated for every node and for t rounds.

Consider an execution where there is a node v with $d' < d$ outgoing edges at round t and note that this node must have had at least one rejected request in each of the t rounds. We can encode the sequence of $tnd\log\Delta$ bits that generates such an execution as follows: The first $\log n$ bits encode node v; The following $t(n-1)d\log\Delta$ bits encode all possible random choices of all nodes but v; As for the random bits of node v, let ℓ_v be the number of random choices actually made by v during the t rounds; we can use

- $2\log\ell_v$ bits to encode in a prefix-free way the number ℓ_v;
- $2\log d'$ bits to encode in a prefix-free way the number d' of accepted requests;
- $\log\binom{\ell_v}{d'}$ bits to encode the positions of the accepted requests in the sequence of ℓ_v requests;
- $\log\Delta$ bits for each one of the d' accepted request and $\log(n/c)$ bits for each one of the $\ell_v - d'$ rejected ones. Notice that we can use only $\log(n/c)$ bits instead of $\log\Delta$ to encode a rejected request since (i) each rejected request was directed to a node that was *overloaded* (i.e., a node that received at least further cd incoming requests) at the time of the request from v; (ii) since each node sends at most d requests at each round, it follows that the number of overloaded nodes is at most n/c at each round; (iii) the previous bits of our encoding uniquely identify the set of overloaded nodes at each round.

In contrast to the $\ell_v\log\Delta$ bits used by node v in the uncompresed encoding we thus use only

$$2\log\ell_v + 2\log d' + \log\binom{\ell_v}{d'} + d'\log n + (\ell_v - d')\log(n/c) =$$

$$= \ell_v\log(n/c) + 2\log(\ell_v d') + d'\log c + \log\binom{\ell_v}{d'} =$$

[12] As remarked in Section 4.2, we can avoid taking ceilings of the quantities measuring the number of bits for the encoding.

$$= \ell_v \log \Delta - \left[\ell_v \log(\alpha c) - 2\log(\ell_v d') - d' \log c - \log \binom{\ell_v}{d'} \right] .$$

Hence, the total number of bits for node v saved in our encoding is

$$\begin{aligned} &\ell_v \log(\alpha c) - d' \log c - \log \binom{\ell_v}{d'} - 2\log(\ell_v d') \\ &\geqslant \ell_v \log(\alpha c) - d' \log c - d' \log \frac{e\ell_v}{d'} - 2\log(\ell_v d') \\ &\geqslant (1/2)\ell_v \log(\alpha c) , \end{aligned}$$

where in the first inequality we used that $\log \binom{\ell_v}{d'} \leqslant d' \log(e\ell_v/d')$ and the last inequality holds for ℓ_v large enough, since c, d', and α are $\mathcal{O}(1)$.

By using that $\ell_v \geqslant t \geqslant (\beta/\log(\alpha c)) \log n$ and that in our encoding we use $\log n$ bits to identify node v, the fraction of strings determining an execution such that there is a node v with $d' < d$ outgoing edges at round t is thus at most

$$\begin{aligned} &2^{-(1/2)\ell_v \log(\alpha c) + \log n} \leqslant 2^{-(1/2)t \log(\alpha c) + \log n} \\ &\leqslant 2^{-(1/2)\beta \log n + \log n} = n^{-(\beta-2)/2} . \end{aligned}$$

□